FabSim Interactive

3.0 for PC

* * * * Semiconductor factory simulator * * * *

****

Help Manual, 12.11.2010

****

Prof. Dr.-Ing. Holger Vogt
Fraunhofer-Institut für Mikroelektronische
Schaltungen und Systeme

Contents

FabSim(TM) is a compact discrete event simulator for (semiconductor) factories. It contains a variety of models for factory setup and wafer flow in a single executable. Interactive simulation provides full control over the virtual fab.

FabSim Interactive Demo (Freely available for download at www.fabsim.com) is limited to up to five tool sets. All other features are equal to the commercial edition.

Contents

1.0 Introduction

FabSim(TM) offers a unique approach to simulate a semiconductor manufacturing plant with discrete-event simulation. As a compact tool in a single executable FabSim allows to simulate a complete factory very efficiently.

Data representing the fab and the planned operation are required as input. No extra effort is needed to set up a factory model. Toolsets comprised of machines are listed with their different recipes, batching capability, MTBF and MTTR. Processes are described by their flow charts which contain a sequence of machines and recipes to be visited by each lot. The lots are started in a planned sequence (start time, process and priority) or using statistical distributions. Alternatively a maximum WIP may be defined. The output is a list of lots leaving the fab after being processed. Other data like toolset usage or buffer occupancy is available as well.

The fab model is set up dynamically at runtime. Up to 128 different toolsets may be used, each set may comprise of up to 50 identical machines (e.g. 15 steppers, 10 furnace tubes, up a total of 6400 pieces of equipment). The number of different process flows is not limited. Any process step sequence may be defined, up to 700 steps per flow.

FabSim Interactive comprises of three files: FabStart.exe is the interactive window. It controls the FabSim simulator which resides in FabSim.dll. FABSIMSTART.HELP is this help file.

2.1 Starting FabSim in batch mode

Like the original FabSim.exe simulator FabSim Interactive can operate in a batch mode. It will read from an input file (e. g. fab_start.bat) any suitable command sequence and act accordingly.

The batch file may contain the following command lines:

```
set FABSIM_IN_DIR=C:\Projekte_Delphi\fabsim_test\in
set FABSIM_OUT_DIR=C:\Projekte_Delphi\fabsim_test\out
FabSim.exe -in_strt lot_sequence.strt -out fabout1 -d  -t 90000
FabSim.exe -in_strt lot_sequence.strt -out fabout2 -d  -t 60000
FabSim.exe -in_strt lot_sequence.strt -out fabout3 -d  -t 130000
```

The first two lines set the environmental variables for the input and the output directories. If not specified, all input files will be searched for only in the subdirectory .\in relative to the directory where FabStart.exe and FabSim.dll reside. By defining the directories as shown above you may set any suitable directory. Besides the absolute path as shown above, relative directory names are allowed as well (e.g. .\in_test).

Lines three to five each specify a complete simulation run. Each line starts with the FabSim executable name (needed by the batch execution of FabSim.exe, but ignored by FabSim Interactive), then any valid command line parameter (as described in chapter 3.3 Starting FabSim) may be added. FabSim Interactive will execute all lines in the batch file sequentially. The simulation currently running is shown in the Box named “Job No.”, starting with “1” for the first simulation run (line three in our example file fab_start.bat) up to “3” in our example file.

On startup FabSim Interactive reads the most recent batch file name from FabStart.ini. This name is displayed in the “Batch file” edit box (first line of the FabSim Interactive window). Before starting the batch simulation you may enter any batch file name manually. A preset file name (fab_start.bat) may be entered by clicking the ‘p’-button. The ‘r’-button will reenter the most recent batch file name stored in FabStart.ini. The ‘n’-button opens a file selection box to search for a new batch file. The “Edit”-button on the right opens the shown batch file with the editor Notepad.exe. If the batch file displayed in the “Batch file” edit box cannot be found in the specified directory on your hard disk, a file selection box opens up. You then may search for the correct batch file to be edited.

The batch simulation is started by activating the “Run Batch” button in the lower left of the window. The small display box “Job. No” below will show the number of the current simulation run.

The edit box “Command line” displays the command line parameters used for the current simulation.

All other (interactive) buttons are deactivated during the simulation.

Output files are found in the directory specified by FABSIM_OUT_DIR. If not specified, output goes to a directory .\out, relative to the directory where FabStart.exe and FabSim.dll reside.

A log file console.log containing comments similar to a console window’s output is written into the directory of FabSim.dll. It may serve debugging purposes.

2.2. Starting FabSim in interactive mode

A major breakthrough is the interactive mode of FabSim. Simulation may proceed for a given amount of time. During the stop following the simulation period the current status can be accessed and lot parameters and factory parameters be changed.

Lot selection

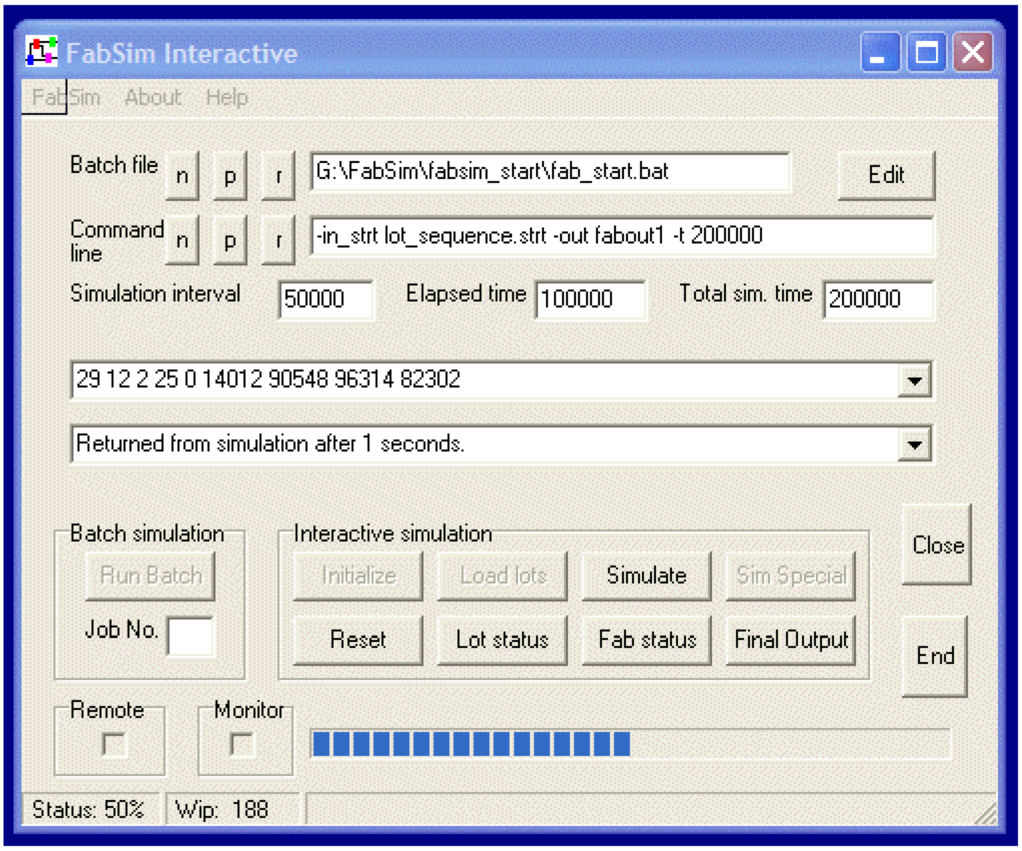
Lots are entered for simulation either from a file specified by the command line parameter following the –in_strt flag (*Command line* box) or are entered manually (-in_strt file.strt not specified!). After each stop a new lot may be entered by specifying its parameters (*New Lot* box) and activating the *Enter Lot* button.

Simulation sequence

- Choose command line parameters.
- Activate *Initialize* button.
- Enter time period for simulation (*Simulation interval* box).
- Activate *Simulate* button (Simulation now starts for the time period specified).
- Select any of the interactive controls, explore status (*Lot Status* and *Fab Status* buttons) and get lots (*Output* button) after simulation has stopped. Optionally choose a lot and step number to halt the simulation during the following simulation period as soon as the step is reached.
- Enter new time period for simulation (*Simulation interval* box).
- Activate *Simulate* button (Simulation now starts the next simulation period as specified).
- Repeat this procedure until total simulation time has been passed (*Total sim. time* box).
- Get final output information and evaluation by activating the *Final Output* button.
- Select *Reset* button to start another simulation (batch or interactive) or *End* button to terminate FabSim Interactive.

With FabSim Interactive you may also start batch type simulation by entering a batch file name into the ‘Batch File’ list box and starting the simulation with the ‘Run Batch’ button.

2.3 FabSim Interactive Main Window



Batch file: G:\FabSim\fabsim_start\fab_start.bat

Command line: -in_strt lot_sequence.strt -out fabout1 -t 200000

Simulation interval: 50000 Elapsed time: 100000 Total sim. time: 200000

29 12 2 25 0 14012 90548 96314 82302

Returned from simulation after 1 seconds.

Job No.

Status: 50% Wip: 188

Edit boxes:

Batch file
Enter a new batch file name manually or by selecting the ‘preset’- or ‘recent’-button. The ‘new’ -button opens a file selection box.

Edit
Opens the batch file set in line ‘*Batch file*’ with notepad.exe. The file may be edited, after saving it’s new content you may start a batch job using *Run Batch*.

Command line
Displays the command line parameters of the current job (batch mode).
Enter command line parameters (interactive mode) manually or by selecting the ‘new’, ‘preset’- or ‘recent’-button. ‘n’ will load a command line from the batch file selected above. If several command lines are stored in the batch file, the final one will be displayed. ‘p’ will offer an internally stored command line, with ‘r’ you select the command line stored in FabStart.ini.

Simulation interval
Enter time period for next simulation phase (interactive mode).

Elapsed time
Simulation time accumulated until current stop (interactive mode).

Total sim. time
Total simulation time chosen by command line parameter –t or preset value (batch mode).
Maximum simulation time chosen by –t parameter or preset value if -t is not specified. Preset value is 200.000 minutes.

Lots returned
Display all lots returned as ready in the simulation period just finished, after button *Output* has been actuated (interactive mode).

Simulation messages
Displays job control commands which have been executed (all modes). If *Remote* is checked, all error messages and warning are also shown here.

Job control and input/output buttons:

Run Batch
Starts batch processing reading the batch file selected in *Batch file* line. Stops automatically when all jobs listed in the current batch file are processed.

Job no
Displays the current job running (batch mode).

Initialize
Initializes FabSim with the data entered in the *Command line* box. Starts the interactive mode. Opens the Interactive Control window.

Load Lots
Opens a new window LoadData which allows to load lots after initialization from file LotStatus.stat.

Simulate
Starts simulation for a time period given in the *Simulation interval* box.

SimSpecial
After initialization this button opens a new Window Simulation Specialties which presents several simulation scenarios (only available in the full version of FabSim). One scenario is described below (Sim BoCont). Another scenario currently under development is a toolset optimization for a given process mix and throughput.

Reset
Frees all memory acquired by FabSim.dll and disconnects dll from master program FabStart.exe.

Lot status
Saves the actual lot status into a text file fab_sim*nn*.lstat. *nn* is the simulation time at the time of saving the lot status data. Data of this file may be used as a starting setup of the fab in a following simulation (see "LoadData").

Fab status
Saves the current factory status into a text file named fab_sim*nn*.fstat. *nn* is the simulation time at the time of saving the factory status data.

Final Output
Add run status information into the output file (*.out) and closes the file. *.exo file will be generated. If -d is selected on the command line, *.log file is readied and closed. If -op is selected on the command line, *.opo with operator info is created.

Close
Sets FabSim Interactive for temination after the current simulation run is finished.

End
Terminates FabSim Interactive immediately. Data may get lost.

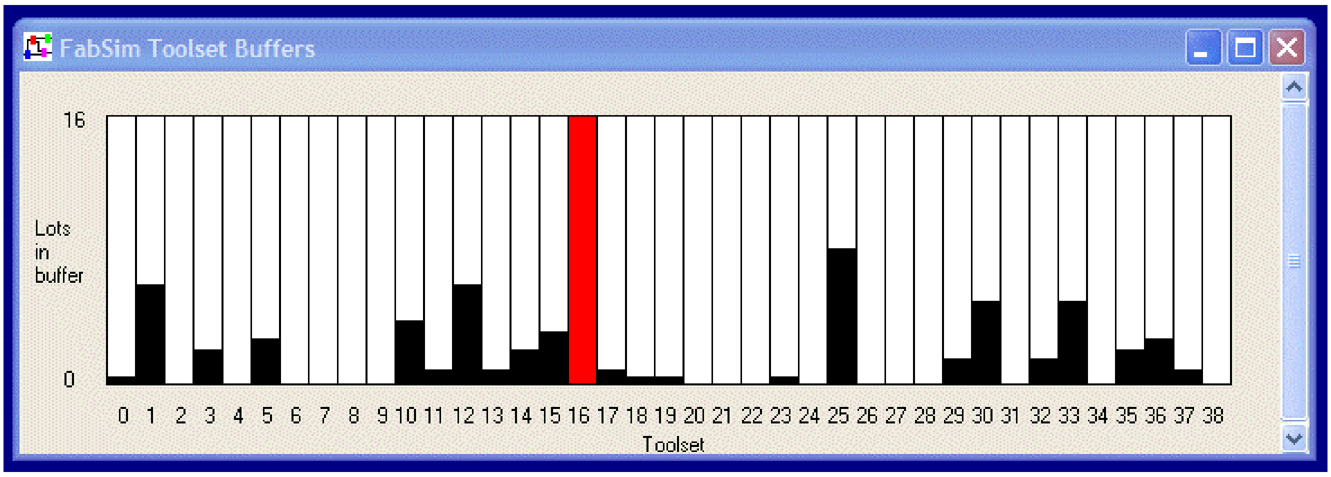
Remote
If checked, FabSim.dll will no longer communicate interactively with the user by (modal) message boxes, but send error codes and error messages directly to FabStart.exe. Error messages will also

be found in *console.log*. It is recommended to start with *Remote* unchecked to get immediate input file error response.

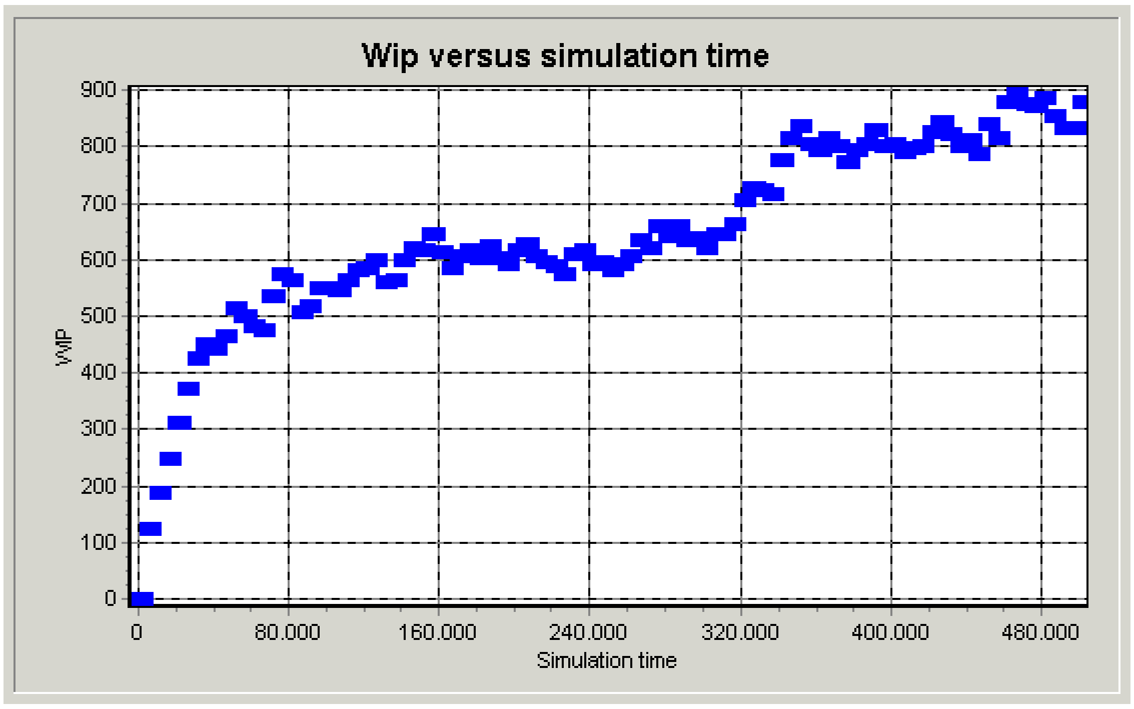
Progress bar
The progress bar displays the status of the current job. A numerical value of the progress status (in %) and the actual WIP are shown in the program's status bar.

Online monitoring:

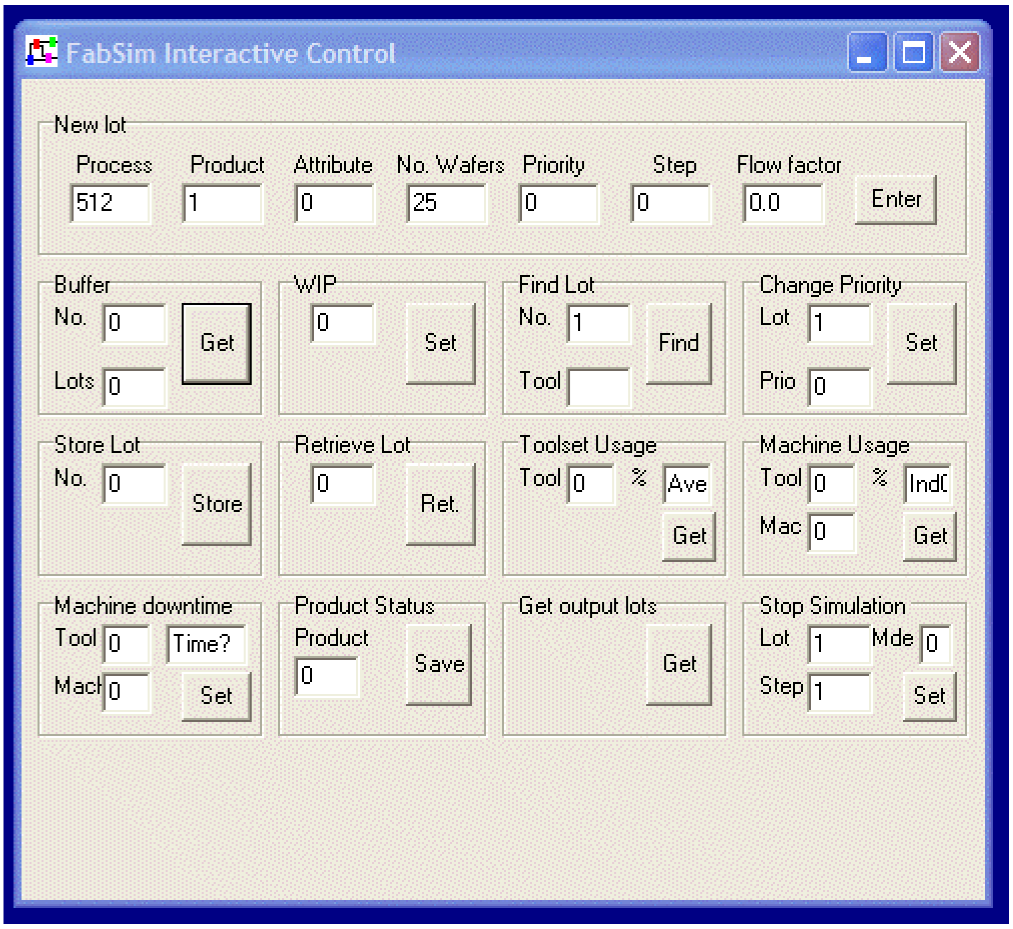
Monitor
If checked, FabSim.dll will open a new window upon initialization. A bar graph plot is started which shows the number of lots waiting in each toolset buffer. The plot is updated dynamically during the simulation. The red bar is the toolset with the largest amount of lots waiting in any toolset buffer. The maximum vertical axis scale (14 lots in the example) is updated according to a running average of the past 10 maxima. The simulation speed is slowed down somewhat due to the plot updates. If the fab is well balanced, the display seems to update erratically, because there is no large buffer. If there is a bottleneck however, it will clearly be demonstrated during the course of the simulation.



A second plot window monitors the progress of the WIP (work in progress) during the simulation. This is a simple and efficient display of the fab stability. If more lots enter the fab than leave it WIP will increase.



Interactive control:


FabSim Interactive Control

New lot

Process	Product	Attribute	No. Wafers	Priority	Step	Flow factor
512	1	0	25	0	0	0.0

Enter

Buffer: No. 0, Lots 0 — Get

WIP: 0 — Set

Find Lot: No. 1, Tool — Find

Change Priority: Lot 1, Prio 0 — Set

Store Lot: No. 0 — Store

Retrieve Lot: 0 — Ret.

Toolset Usage: Tool 0 % Ave — Get

Machine Usage: Tool 0 % Ind(, Mac 0 — Get

Machine downtime: Tool 0 Time?, Mach 0 — Set

Product Status: Product 0 — Save

Get output lots: Get

Stop Simulation: Lot 1 Mde 0, Step 1 — Set

New lot

Enter new lot (one lot at each stop). Process, product, number of wafers and priority are entered as integer numbers (interactive mode).

Enter
Enters lot specified in the *New lot* edit boxes (one lot per simulation stop).

Buffer
Shows number of lots (*Lots* box) waiting in a buffer in front of the toolset specified (*No.* box). Buffer no. and max. buffer size is entered for BoCont simulation.

WIP
Changes maximum work in progress (WIP).

Find Lot
Displays toolset (*Tool* box) where lot is actually waiting or in process.

Change Priority
Changes priority (*Prio* box) of lot selected (*Lot* box).

Store Lot
Puts lot (*No.* box) on hold by sending it to a common buffer.

Retrieve Lot
Retrieves lot (*No.* box) and reenter it into processing.

Toolset Usage
Retrieves average toolset use (process time versus total time in %) since the most recent start or restart of the simulation.

Machine Usage
Retrieves machine use (process time versus total time in %) since the most recent start or restart of the simulation.

Machine Downtime
Sets forced downtime of a specific machine in a given toolset. The machine will go down after the lots it is currently processing have left. If the machine was empty, it will go down immediately. During a stop several machines may be set to down.

Product Status
Saves the current lot status for a given product into a text file named fab_sim*nn*.pstat. *nn* is the simulation time at the time of saving the product status data. The product name is entered into the edit box.

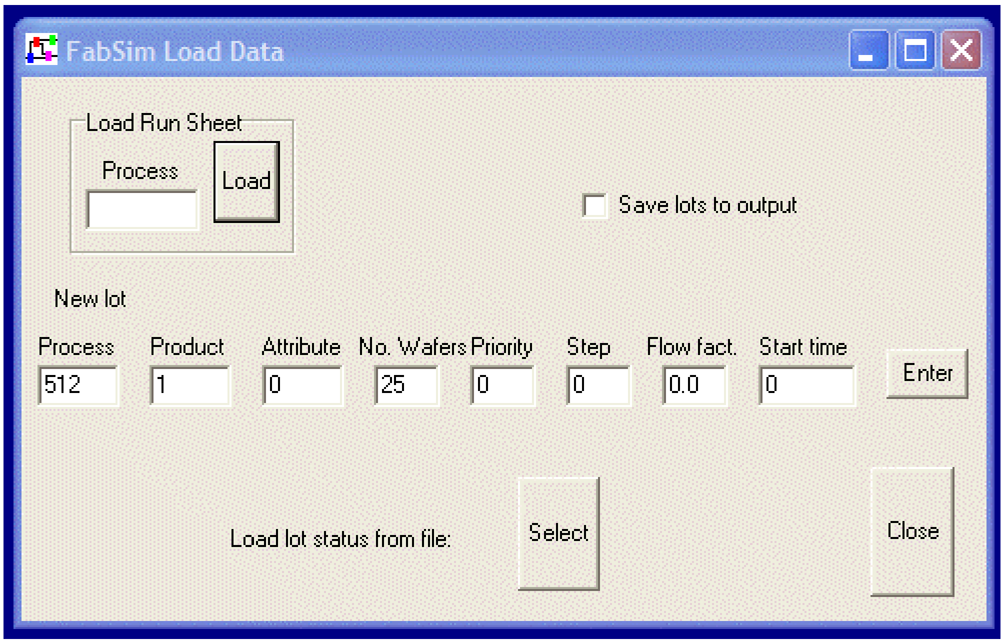
Get Output Lots
Retrieves all lots readied in the simulation period just finished. Lots will be displayed in the *Lots returned* pull down box of the main window.

Stop Simulation
Stops the simulator according to the Mode entry (1, 2, or 3) and the input data. FabSim then waits for input and may be continued. After entering the mode, the input box descriptions change according to the further entries required.

Mode 0: No stop
Mode 1: FabSim stops when the given lot reaches the given step.
Mode 2: FabSim stops whenever a lot with the given process reaches the given step.
Mode 3: FabSim stops, when the buffer size (number of lots in the buffer) of the given toolset exceeds the given number (size > 0). You may enter size as a negative number, then FabSim stops as soon as the buffer size undercuts the given size (it’s absolute value).

LoadData

After initialization of FabSim.dll lots may be entered before simulation commences. You may either load several lots manually or read in lot status information from a file LotStatus.stat.



Process	Product	Attribute	No. Wafers	Priority	Step	Flow fact.	Start time
512	1	0	25	0	0	0.0	0

Load Run Sheet
After entering a valid run sheet name (integer number), "*Load*" will load the run sheet into FabSim.dll.

New Lot
After entering valid lot data (integer numbers for: process name, product id, number of wafers, priority, starting step (any step of given run sheet), flow factor, start time) "*Enter*" will load the lot into FabSim.dll, directly into the buffer of the toolset which is responsible for processing the given step.

Load lot status file
"*Select*" button opens a file dialog where file fab_sim*nn*.lstat saved after a previous simulation will load its currently active lots into the buffer of the respective toolset. FabSim.dll then starts with these lots already in the fab. This serves either to skip a simulation stabilization time. On the other hand FabSim Interactive may be used to do short term forecasts starting with the actually loaded fab. If "*Save lots to output*" is not checked, all lots thus loaded will be stored in *.scrap so not to disturb output data of lots started regularly. If checked, all lots ready are saved in *.out.

Close
The modal window "LoadData" has to be closed before simulation may be started.

Simulation Specialties

This window is only available in the full version of FabSim.

Sim BoCont
Emulates bottleneck control by reducing the lot start frequency to 50% if the number of lots in the selected buffer is exceeded. This procedure is as an example implemented into the supervisor program FabStart.exe. FabSim has a similar control mechanism (no lot start if buffer size limit is exceeded) using the –buf command line parameter. Using *BoCont* the input data are generated as follows: Total simulation time is read from '*Total sim. time*'. The simulation is interrupted every time step read from '*Simulation interval*'. During each interrupt a new lot as set in '*New lot*' is started.

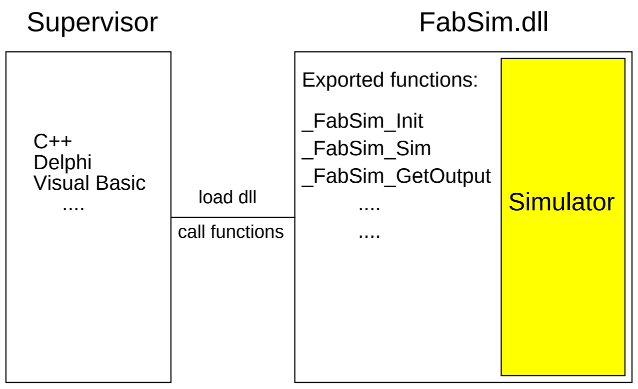
Buffer selection and buffer size limit are set in ‘*Buffer No.*’ and ‘*Buffer Lots*’. ‘*Command line*’ should not contain the –in_strt option. Simulation is started by *Initialize* and then *BoCont*.

Simulated Annealing
A scenario currently under development is a tool count optimization procedure for a given process mix and throughput.

2.4 FabSim Interactive Interfaces

FabSim.dll

FabSim Interactive is located inside a Windows dynamic link library FabSim.dll. Such a dll is an executable which cannot run on itself, but is loaded at runtime into another executable, e.g. the supervisor FabStart.exe. There it offers its exported functions to the supervisor. The functions may be called as if they were defined directly inside the supervisor program. FabSim.dll export more than 25 functions. They may be used to control the simulator, thus allowing interactive simulation.



In interactive mode of FabSim we firstly initialize FabSim by loading the dll and reading all factory input data. An empty factory or the factory status saved during a previous simulation may serve as a starting point. A lot may now be entered. The supervisor then starts FabSim.dll for at least one clock cycle. After this simulation period has passed, FabSim stops and waits for input. All data are stored and are accessible by several of the exported functions. You may search for lots, save the complete lot or fab status, enter a new lot, move a lot out of the process flow and onto a shelf, retrieve another lot from the shelf for further processing, set machines down for a given period, change lot priority, ask for toolset and machine utilization. You may even toggle from the push mode (each lot leaving a toolset will find its way into the buffer and then into an idle machine of the following toolset) into a mode where the supervisor actively has to move each lot from the toolset buffer into a machine currently available. Thus the supervisor has full control over the simulation procedure. During periods where FabSim.dll is allowed to run uninterrupted, it will retain the same high simulation speed as the command line version.

Instead of using the supervisor program FabStart.exe, you may write your on simulation controller, add other GUI interfaces or integrate FabSim.dll into a larger factory control framework. Detailed interface specifications, e.g. descriptions of the functions included their parameters are summarized in another document available with the full version of FabSim.

Cdispatch.dll

FabSim or FabSim Interactive itself may now call a dll called Cdispatch.dll. Inside of this dll you define your own dispatching rules, that is you may determine the reasoning for sorting the buffer in front of each toolset. For this purpose Cdispatch.dll has full access to toolset data, as offered in mach_dat.mset, to all flow charts, to lot data and to all toolset buffers. These data are then used to define customer specific rules to select the lot which will be processed next in the specific toolset. Each toolset may have it’s own rules. A mixture of internal rules and customer defined rules is possible. The C++ source code of the Cdispatch.dll frame is delivered with the full version of FabSim. After coding the dispatch rules and compiling the dll, FabSim now emulates any factory behavior. Experiments running different rules are done quickly, there is only a small simulation time overhead to be paid for. Local dispatching rules as well as look ahead or look back rules, minimum inventory variability scheduling procedures and others may be developed with only minor effort. As examples, two rules are already defined, a priority rule based on the attribute parameter of each lot or a rule searching for the toolset with the smallest buffer following the current process step. Only the lot heading towards this toolset is selected for the current process step.

3 FabSim

3.1 Program Features

FabSim is a discrete event simulator designed to simulate a (semiconductor) factory. Machines may be set up, lot sequences (run sheets) defined and lots started at any time during the simulation flow. Outputs will be lots processed versus time and many other parameters.

CPU time for simulation is a few 10 seconds up to a few minutes per year fab time. It strongly depends on the size of the fab simulated. A pilot line may simulate in 30 seconds, a 500 wafer starts per day fab in 5 minutes (1.4 GHz Athlon, LINUX).

Including operators into the simulation has been prepared, but this features is not yet released in the current program version.

Inputs:
- available tool sets (128 with up to 50 equal machines each)
- matrix of transport time between toolsets
- lot start sequence
- run sheets

Outputs:
- lots processed
- lots scrapped
- machine usage
- buffer occupation
- machines available (not defect)
- operators required
- detailed log of each process step
- (optional debug info)

Interface to Cdispatch.dll for customer defined dispatching rules (FabSim Interactive)

3.2 Setting up FabSim

1. List all equipment used for processing. Up to 128 tool sets (having up to 50 equal copies of a machine, here sometimes called subunits or subs) may be installed. For each tool set up to 9 process recipes may be defined. An important parameter is the wafer processing time of each recipe. Put all these data into the file mach_dat.mset (see section 3.4.2).

2. Define (multiple) flowcharts as a sequence of process steps. You will set up one file per flow chart (see section 3.4.4). The maximum number of steps in a flow chart is set to 700.

3. Determine the lots to be started by adding start time, process flow and some other parameters to the file lot_sequence.strt (see section 3.4.1).

4. Start FabSim per console command line or from a batch file (see also chapter 2.1). The batch file (e.g. fab_start.bat) allows to specify individually named input and output files, so a batch job with several parallel or consecutive simulations is possible.

5. Evaluate the simulation data by: a) looking at output files, b) use spreadsheet (EXCEL) macros delivered with the simulator or c) define your own data processing using the FabSim ASCII output files as input. (see also chapter 3.6)

6. The simulation time resolution of FabSim is determined by your input data. All time data (process and downtime duration, start of lots, simulation time and others) are given as multiples of an internal simulation clock. For a semiconductor fab a 1 minute resolution is appropriate. Typical simulation runs will take 200.000 to 500.000 cycles (approx. 0.4 to 1 year fab time). If you need more details, you may specify all time periods in multiples of 0.5 or 0.1 minutes or even in seconds. Of course simulated fab time will decrease accordingly if you do not increase the number of clock cycles (-t *nnn* on the command line).

3.3 Starting FabSim

Start FabSim with optional command line parameters. Sometimes a numerical or text value following a command line parameter is required (e.g. [-t time1]), sometimes it is optional (e.g. [-trace [machine no.]]).

FabSim [-d [toolset no.]] [-t time] [-p1] [-op] [-in_op filename] [-use starttime]
[-in_strt filename] [-in_mset filename] [-in_fto filename] [-out filename]
[-in_dir dir] [-out_dir dir] [-trace [toolset no.]]
[-ff0 value] [-ff1 value] [-ff2 value] [-sptf_wait time] [-batch_wait time]
[-lot_y percent] [-waf_y percent] [-wip no_lots] [-buf buffer_no no_lots]
[-rand_dur percent] [-rand_unl time] [-exp_dur] [-set_rand value] [-cost]
[-all_cr] [-all_sptf] [-all_odd] [-all_edd] [-all_wspt] [-all_dispa value]
[-no_out] [-cont_out] [-ga_mtbf [value]] [-ga_mttr [value]]

Debug options may be added to the command line:
-no_mttr -no_maint -no_delay -no_setup -no_trans

All command line parameters may be added to the command line (on the console or in a batch file) in any order.

If '**-d**' is chosen, a file fab_sim.log (or *.log with * given after -out) will be generated with runtime information on machines used or defect. In addition if -wip is set, lots not started due to excess wip count are listed. Optional machine no. will restrict the log output to the toolset with number "toolset no.".

'**-t**' followed by time1 sets the total simulation time in minutes. If '-t time1' is not given, simulation time will be 200.000 minutes.

'**-p1**' will allow priority 1 simulation, however simulation time will increase by 30% to 50%. Prio 1 lots will always run ahead of all other lots and block the next machine in advance until being processed. If '-p1' is not selected, prio 1 lots will move to the front in each machine queue (They will be started before lots with priority 2, finally prio 0 lots will follow.).

'**-op**' generates an output file *.opo which lists the number of operators required for the given process mix and throughput. The input file operators.in (or *.in if -in_op is set) is used to enter some operator data needed for the calculation.

'**-in_op**' with the filename following immediately will include operator data into the simulation. It replaces the standard operator input file operators.in.

'**-use**', followed by time2 defines the point from which on the machine use factor and tabulated cycle time data (Eval_ct.xls) will be collected. You may choose time2 so that data collection starts only after the simulation (e.g. the wip in the factory) has stabilized. If not specified, data collection will start immediately.

The filename given after '**-in_strt**' replaces the default input file 'lot_sequence.strt'.

The filename given after '**-in_mset**' replaces the default input file 'mach_dat.mset'.

'**-in_fto**' sets the file name for the from-to matrix (default: 'mach_dat.fto').

The string given after '**-out**' replaces the names for the default output files 'fab_sim.out', 'fab_sim.use', 'fab_sim.log', 'fab_sim.tra', 'fab_sim.occ', 'fab_sim.wip'. If for example the command says -out fabnew, fabnew.out is generated instead of fab_sim.out.

'-in_dir', '-out_dir' followed by a path to a directory, will set the input and output directories. Data given here will override all preset values and the environmental variables FABSIM_IN_DIR and

FABSIM_OUT_DIR. Any absolute (e.g. F:/fabsim/in) or relative (e.g. ./fabdir/out) path is valid.

'**-trace**' will generate tracing by three output files in an EXCEL readable format ('fab_sim.tra', 'fab_sim.occ', 'fab_sim.wip'). A toolset number following –trace will restrict tracing to the toolset specified. This might be useful if during long simulation runs very large trace files are generated.

'**-ff0, -ff1, -ff2**' will set the flow factor for all lots with priorities 0, 1, 2 respectively and override the internally preset values.

'**-sptf_wait**' overrides the preset maximum waiting time (720 min) of lots in the buffer under SPTF or WSPT rule. If a lot has been waiting for longer than set in this option, it will move to the front of the buffer queue. That is, it will be started when a machine is available, even if it is currently not the lot with minimum processing time.

'**-batch_wait**' sets a maximum waiting time of lots in the buffer of a batch system. If a lot has been waiting for longer than set in this option, it will be started as soon as a machine is available, even if currently the minimum batch size has not been reached. Then a batch with less than min_ba lots will be formed.

'**-lot_y**' sets the percentage of lots leaving the fab ready versus lots started. With random numbers lots are selected to be scrapped as well as the scrap step number of each lot. Scrapped lots are assembled in file *.scp.

'-waf_y' allows to define the number of wafers coming out of the fab versus wafers started (in percent). Per random number generator lot size is reduced according to the percentage given.

'**-wip**' followed by 'limit_lots' will limit the number of lots in the fab to 'limit_lots'. The next scheduled lot will not be started and skipped if the number of active lots would otherwise exceed 'limit_lots'. This control procedure is often called CONWIP (constant wip).

'**-buf**' followed by 'toolset number' and 'maximum number of lots' allowed in the selected buffer will limit the number of lots released into the fab. Only when the buffer size limit is not exceeded, lot starts are allowed. Otherwise the lots due will be skipped and not started. This control procedure may be called CONBUF. Especially selecting the bottleneck buffer is of interest.

'**-rand_dur**' followed by 'percent' adds (or subtracts) a random variation to the process step duration for all toolsets. This option may be used to add further variance to the simulation. A gaussian random number with standard deviation set in percent of recipe duration will be added (or subtracted) to the preset recipe duration.

'**-exp_dur**' provides an exponential distributed process step duration for all toolsets. The mean value is given by the data rec0 to rec8 provided in 'mach_dat.mset'.

'**-rand_unl**' followed by 'time' sets a random unload delay. This option may be used to emulate operator behavior if processing is limited by operator availability. If no operator is available, a machine will not be unloaded, then of course a new lot cannot be started. This is equivalent to adding time steps to the unload time. A random number with exponential distribution and mean set by 'time' will be added to the unload duration. Priority 1 lots are exempt because they are served by 'hand carry'.

'**-set_rand**' sets a seed value for all random number generators. FabSim uses several random number generators, e.g. for determining time of machine failure, repair time, random variability of process duration, waiting time, rework. Homogeneous, exponential or normal distributions of random numbers are used. All random number generators generate sequences of pseudo random values. Within a sequence the numbers are approximately random. The sequence itself is however determined by a seed value offered to the generator. That is, the sequence will be repeated if the generator is started with the same seed value. If you start FabSim without setting –set_rand *nn*, a seed value will be determined by the program depending on the system time of the computer. So every FabSim run will result in a different output, because system time has progressed, a different seed is chosen, and thus all random variables may become different. If –

set_rand *nn* (withh *nn* >= 0) is set, *nn* will lead to a fixed seed value for all random number generators. Repeating FabSim (without changing anything else) will now always lead to the same output data.

'**-all_cr**' will set all toolsets to use the 'critical ratio' (CR) dispatching rule. This command will override the toolset buffer status given in column 'dispa' of toolset input file *.mset.

'**-all_sptf**' will set all toolsets to use the 'shortest processing time first' (SPTF) dispatching rule. This command will override the toolset buffer status given in column 'dispa' of toolset input file *.mset.

'**-all_odd** will set all toolsets to use the 'operation due date' (ODD) dispatching rule. This command will override the toolset buffer status given in column 'dispa' of toolset input file *.mset.

'**-all_edd** will set all toolsets to use the 'earliest due date' (EDD) dispatching rule. This command will override the toolset buffer status given in column 'dispa' of toolset input file *.mset.

'**-all_wspt** will set all toolsets to use the 'weighted shortest processing time' dispatching rule. The lots attribute is chosen as the 'weight'. The lots are sorted according to *'process step time of next machine'/weight*. After defining the weight, the wafer count of each lot is set to 25 internally. Thus no wafer count dependent process step time is take into account. Each priority level has it's own queue in the toolset buffer, which will be sorted according to WSPT. Lots from prio 1 queue are chosen first for processing, prio 2 comes second and finally lots without priority are processed. -sptf_wait *nn* may be set here too to define a maximum waiting time *nn* for each lot. If it has passed, the lot is moved to the front of the queue disregarding it's wspt value.
'**-all_dispa** followed by an integer number *nmm* will set the dispatching rules as a combination of internal rule *mm* and an external rule *n* as described for mach_dat.mset.
'**-no_out**' will suppress writing the output to the *.out file (After the simulation has finished, *.exo will be generated anyway). In FabSim Interactive this command also prevents writing runtime information to console.log. Therefore this option should only be used with a proven input data set.

'**-cont_out**' will allow to write data to the output to the *.out file during simulation. This slows down the simulator if a large data volume is written. Per default *.out is generated after the simulation has finished (automatically with FabSim.exe or in batch mode of FabSim Interactive, after pushing the GetFinalOutput button in FabSim Interactive).

'**-ga_mtbf**', followed by an optional real number, typically less than 1, say 0.6, will set all mtbf selections to draw from the Gamma distribution (instead of exponential). The shape factor *alpha* of the Gamma distribution is set to the real number. If no number is entered after -ga_mtbf, alpha = 0.7 is assumed. The scale factor *beta* is set to the mtbf value read from mach_dat.mset. You may overwrite this statement locally for the given toolset (see chapter 3.4.2).

'**-ga_mttr**', followed by an optional real number, typically larger than 1, say 1.2, will set all mttr selections to draw from the Gamma distribution (instead of exponential). The shape factor *alpha* of the Gamma distribution is set to the real number. If no number is entered after -ga_mttr, alpha = 1.3 is assumed. The scale factor *beta* is set to the mttr value read from mach_dat.mset. You may overwrite this statement locally for the given toolset (see chapter 3.4.2).

Several command line options allow quick fab model debugging: **-no_maint**, **-no_mttr**, **-no_delay** , **-no_trans**, **-no_setup** will suppress scheduled or unscheduled machine downtime, remove load and unload delay times, set transport time to 0 and suppress machine setup, regardless of any entry to the *.mset input file

All parameters selected have to be integers (except ff0 to ff2 and parameters to -ga_mtbf, -ga_mttr being real values) or valid file names.

With LINUX OS replace FabSim by ./FabSim.x on the command line to start the application.

************************ batch simulation (see also chapter 2.1)*******************************
You may assemble several command lines as described above into a single batch file. The file shown below (LINUX batch script) has been used to calculate throughput versus WIP for a 500

wafers starts per day fab. Under Windows you may call it fab_start.bat and execute it by typing fab_start onto the command line. Under LINUX you additionally have to make this file executable and call it by ./fab_start. Output files have been set differently for each simulation run. -wip has been varied as the parameter of investigation. Simulation time is 500.000 minutes (approx. 1 year). Machine occupancy will be evaluated in the stabilized phase after 300.000 minutes.

```
./fabsim.x -t 500000 -in_strt proc025.strt -in_mset EL025HV.mset -out n1400 -use 300000 -wip 1400 > fab_start.log 2>&1
./fabsim.x -t 500000 -in_strt proc025.strt -in_mset EL025HV.mset -out n1200 -use 300000 -wip 1200 >> fab_start.log 2>&1
./fabsim.x -t 500000 -in_strt proc025.strt -in_mset EL025HV.mset -out n1000 -use 300000 -wip 1000 >> fab_start.log 2>&1
./fabsim.x -t 500000 -in_strt proc025.strt -in_mset EL025HV.mset -out n900   -use 300000 -wip 900 >> fab_start.log 2>&1
./fabsim.x -t 500000 -in_strt proc025.strt -in_mset EL025HV.mset -out n800   -use 300000 -wip 800 >> fab_start.log 2>&1
./fabsim.x -t 500000 -in_strt proc025.strt -in_mset EL025HV.mset -out n700   -use 300000 -wip 700 >> fab_start.log 2>&1
./fabsim.x -t 500000 -in_strt proc025.strt -in_mset EL025HV.mset -out n600   -use 300000 -wip 600 >> fab_start.log 2>&1
./fabsim.x -t 500000 -in_strt proc025.strt -in_mset EL025HV.mset -out n500   -use 300000 -wip 500 >> fab_start.log 2>&1
./fabsim.x -t 500000 -in_strt proc025.strt -in_mset EL025HV.mset -out n400   -use 300000 -wip 400 >> fab_start.log 2>&1
./fabsim.x -t 500000 -in_strt proc025.strt -in_mset EL025HV.mset -out n300   -use 300000 -wip 300 >> fab_start.log 2>&1
```

3.4 Input Files

3.4.0 Where to locate input and output files?

All files giving input to FabSim.exe are per default expected to be in the same directory as the program file FabSim.exe (FabSim.x). All output file will appear there too. FabSim Interactive locates input files in a default directory ./in (where "." denotes the program file directory), output files are written to ./out). If one of the required input files or the input directory are not found, an error message appears. If FabSim Interactive does not find the ./out directory, it will generated one automatically, a message goes into the console.log file.

If output files should go into a different directory, you may set the environmental variable FABSIM_OUT_DIR, in LINUX (bash) e.g. with export FABSIM_OUT_DIR=/usr/local/fabout. You may remove this setting with unset FABSIM_OUT_DIR. The length of the path + filename is limited to 150 characters.

Input files may be found in another directory, e.g. in /usr/local/fabin, if FABSIM_IN_DIR is set by export FABSIM_OUT_DIR=/usr/local/fabin (LINUX bash). You may remove this setting with unset FABSIM_IN_DIR. export FABSIM_IN_DIR= will not work because FABSIM_IN_DIR will not be removed, but stays empty.

The equivalent commands in the Windows DOS-box or the fab_start.bat script are: set FABSIM_OUT_DIR=c:\fabsim/out for adding the environmental variable, set FABSIM_OUT_DIR= for removing it.

If you start FabSim Interactive in batch mode, you may use the same DOS-compatible commands to set the in and out directories in the batch file. Repeated use of these commands is possible.

Finally the in and out directories may be set via command line parameters -in_dir or -out_dir. Either command has to be followed by a valid path eiter absolutely, e.g. beginning with C:/ or by a path relative to the FabSim executable's directory. -in_dir and -out_dir will override the preset paths and the paths set via environmental variable, as described above.

It is possible to load an individual input file from another directory from the command line. For example -in_mset ./inin/mach_dat.mset will search for mach_dat.mset in a subdirectory /inin located relative to the input directory set via the environmental variable or command line. -in_mset ../inin will search for mach_dat.mset in the directory tree moving up one directory and then down to /inin, starting again from the input directory set via environmental variable or command line.

3.4.1 *** lot_sequence.strt ***

The file contains the sequence of lots to be started. Lots may be started in a deterministic sequence as shown in section 3.4.1.1. It is also possible to start FabSim with input data generated from statistical distributions. Start lot interarrival times, number of wafers in a lot and priority are chosen randomly. The format of lot_random.strt is described in section 3.4.1.2.

3.4.1.1 Deterministic lot start

First example file:

```
title: 0.25um, double loop test
relative
number: proc:  prod: attrib: wafers:  start:  prio:   flowf:
start_loop
1       025    1     0       12       90      0       0
repeat 2000
1       012    3     0       25       70      1       2.5
start_loop
1       025    2     0       25       80      0       4
1       012    3     0       25       75      0       4
repeat 2500
1       025    2     0       18       70      2       0
```

Second example file:

```
title: 0.25um, no loop
absolute
number: proc:  prod: attrib: wafers:  start:  prio:   flowf:
1       025    3     3       12       90      0       0
2       012    1     2       25       180     1       0
3       025    2     2       25       270     0       0
4       012    1     3       18       360     0       0
5       012    1     1       25       400     2       0
```

Third example file:

```
title: 0.25um, mixed single lots and loops
absolute
number: proc:  prod: attrib: wafers:  start:  prio:   flowf:
start_loop
1       025    3     0       12       90      0       0
repeat 100
101     012    1     3       25       9100    1       0
start_loop
3       025    2     4       25       60      0       0
4       012    1     56      18       120     0       0
5       012    1     4523    25       180     2       0
repeat
```

The file starts with a title line. It may contain any title or comment describing the file. The second line contains the keyword "relative" which sets relative lot start times and lot numbers. Stating "absolute" will allow to start lots outside loops with exact start time and lot number given. Both keywords “relative” and “absolute” provide a deterministic lot start sequence.

Lots are described by a set of integer numbers (except for the last number, which is of real type) separated by a tab (or blanks) like "1 12 3 437 18 70 2 2.3". The numbers have the following meaning:

1	number:	lot number
12	proc:	process name
3	prod:	product name
437	attrib:	an attribute to the lot which may be selected freely
18	wafers:	number of wafers started
70	start:	start time
2	prio:	priority (0, 1 or 2)
2.3	flowf:	flow factor

Prio 0 means no priority at all. Prio 2 lots will move to the front of each toolset queue. If option '-p1' is selected in the command line, Prio 1 lots will make machine reservations and are processed by 'hand carry'. If the option '-p1' is not selected, priority 1 lots simply move to the front of each toolset queue, in front of Prio 2 lots.

The flow factor determines the due time for each lot. Due time is calculated by multiplying the raw processing time (All process step times read out from the toolsets as requested by the run sheet are added.) with the flow factor. The flow factor for each lot may be set according to the following hierarchy: The value in lot_sequence.strt in the column flowf: overrides all other settings for the specific lot. If this value is set to 0, then either the preset values are valid: 6 for all prio 0 lots, 4 for all prio 2 lots, 2.5 for all prio1 lots. If however you specify '–ff0 5.4' on the command line, this value will override the preset value for all prio 0 lots. You may also override the preset flow factor values of prio2 or prio1 lots by setting '-ff2 *m.n*' or '-ff1 *m.n*' respectively.

The attribute is useful to enhance priority lot selection in cooperation with the customer API Cdispatch.dll. It is also used as weight in the WSPT dispatching rule.

The minimum start time given in column *start:* is 2 due to some set ups made in the first simulation cycle of FabSim.

Lots may occur individually or within loops. Each loop begins with "loop_start" and ends with "repeat". Any number of lot lines may be set into a loop between "start_loop" and "repeat".

"repeat *nn*" will allow to run through the loop *nn* times and then proceed to the line following "repeat *nn*". The final loop may use "repeat" without "*nn*". This loop will be repeated for ever (until the simulation time given by command line option -t has passed). "repeat" without "*nn*" may of course be applied in the first loop if you wish to have one loop only.

The first example file starts with a first loop. 2000 times a lot with the data given by "1 025 1 12 90 0" will be started. The lot numbers from column *number:* are not used inside the loop. Instead each new lot will get its own number, each time increased by 1. The first lot will start at time 90. 90 is then used as relative offset. So the next lots start at 180, 270, 360... After 2000 lots the loop finishes. Next a single lot of type 1 012 3 25 70 1 will be started. Its lot number will be 2001. Due to "relative" stated in line 2 also for an individual lot the lot number and the start time are set relative to its predecessor. The next loop is repeated 2500 times. Its "loop start time" is the time the previous lot has commenced. Two lots with different processes are started within the loop. The first lot is entered at "loop start time" plus offset 80, the next at 80 + 75. Then the loop repeats, the new loop start time is set to the start time of the previous lot. Finally a single lot will be started.

In the second example file five lots are started. Line two states "absolute". Lot numbering is automatic, column *number*: is ignored, but start time is set exactly to the values given in column *start*: .

The third example file contains two loops with an individual lot in between. The lot number is again set automatically. Line 2 states "absolute". Thus the start times of the individual lot and of all lots inside a loop with multiple lots have to be chosen carefully. Inside the loop the start time now is calculated by storing the loop start time, then adding the start: value of each lot given to the loop start time. The lots listed outside a loop will be started exactly at the time given in column *start:*. Choosing a wrong start time will lead to skipping the lot. This will happen for example if the start time for the lot is set to a value smaller than the actual simulation time. If the individual lot's start time is set to 9000 minutes in the sample file, this time will have passed when trying to start the lot because the first loop already ran for 9000 minutes. The first lot starts at 90, the final lot of the 100 lots started in the first loop starts at 9000. The individual lot starts at 9100, the first lot in the second loop at 9160, the second at 9220, the third at 9280. This value is now the new loop start time.

****

3.4.1.2 Lot start data drawn from statistical distributions

The keyword “random” in line 2 of lot_sequence.strt serves to select a lot start sequence, whose values interarrival time, lot size and priority are drawn from statistical distributions via random number generators.

Line 1 is a title of the file, not used by the program. If row 2 states “random” the line format following row no. 3 is now different from what has been described in section 3.4.1.1. Each new line will be the source of a sequence of lots. All these lots are assembled into an input sequence and provided as input to FabSim.

```
title: small fab, random choice of interarrival time, lot size and priority
random
proc  prod att_min att_max flowf  waf_min  waf_max  prio_mi  prio_ma
intarr
512   45   1       5       3.5    20       25       0        2        2100
12     7   1       5       3.5     3        7       0        2        2300
```

After entering the process, product and flow factor, *waf_min* and *waf_max* set the lot size. Each lot generated by this row will have its wafer count between and including *waf_min* and *waf_max*. The exact amount of wafers for each lot is drawn from a random number generator with uniform distribution between these limits.

Prio_mi and *prio_ma* are the limits of the priority value drawn from a uniformly distributed random integer number.

Lots follow each other with a time interval called interarrival time. Interarrival time will be drawn from an exponential distribution. *intarr* is the mean of this distribution.

The example file shown above will lead to a lot sequence which listed here:

```
title: small fab, random choice of interarrival time, lot size and priority
absolute
number: proc:   prod:   attrib: wafers: start:  prio:   flowf:
1       512     45      3       25      4       1       3.5
2       12      7       3       7       5       1       3.5
3       512     45      1       25      117     1       3.5
4       12      7       5       7       128     1       3.5
5       512     45      3       20      159     0       3.5
6       12      7       4       3       174     0       3.5
...
...
447     12      7       3       3       485104  2       3.5
448     512     45      2       24      485823  2       3.5
449     512     45      5       20      486417  2       3.5
450     12      7       1       4       488921  2       3.5
451     12      7       5       3       492133  1       3.5
452     512     45      4       25      493404  1       3.5
453     512     45      4       25      493468  0       3.5
454     512     45      2       21      493750  2       3.5
455     512     45      1       20      496203  0       3.5
```

The two input rows generate lots according to the data given until the maximum simulation time (500000 minutes) has passed. FabSim Interactive will write the lot list to console.log, FabSim.exe will display it on the console. You may redirect it into a file by the > operator for documentation.

3.4.2 *** mach_dat.mset ***

Toolset and machine data are stored in mach_dat.mset or the file name following the command line parameter -in_mset.

The total number of toolsets is listed in line 2. The current program version allows a maximum of 128 different toolsets (numbered 0 to 127). Each toolset may contain up to 50 equal machines (parameter machine), e.g. 15 poly etchers or 20 gate oxide furnace tubes.

Each line following line 3 is organized according to the headers given in line 3:

example:

```
Total number of machines:
2
toolset machine recipes max_ba min_ba dispa  load_d op_load unl_d op_unl
0       1       1       2      1      1      5      0       5     0
1       1       3       2      1      0      5      0       5     0
```
(to be continued:)
```
transp op_tran mtbf   mttr  op_mttr stbm1  sttm1 op_mt1 stbm2   sttm2
10     0       10000  1000  0       10080  120   0      259200  1440
10     0       10000   500  0       10080  120   0      259200  1440
```
(to be continued:)
```
op_mt2 min_wt setup op_set setupt1 setupt2 int_d op_proc rec0 rec1 rec2
0      0      0     0      0       0       0     151020   50   0    0
0      0      0     0      0       0       0      22100   10   15   26
```
(to be continued:)
```
rec3 rec4 rec5 rec6 rec7 rec8 comment
0    0    0    0    0    0    RCA cleaning
0    0    0    0    0    0    wet etch 1,2,3
```

toolset: number of the tool set, starting with 0. Numbering should be consecutive, otherwise an error might occur.

machine: number of equal machines per tool set as described in the respective line, e.g. three equal furnace tubes, ten equal sputter coaters etc.

recipes: each machine may have up to 9 different recipes. Within a tool set all machines have the same recipies. A process may choose any of these recipes.

max_ba: Batch operation of machines is determined by max_ba and min_ba. Batch machines assemble multiple lots, which request the same process step, into batches. A batch is then processed in a single step. After processing is finished, the batches are disassembled again into the individual lots.

If max_ba is larger than 1, but smaller than 1000, it sets the maximum number of **lots** possible in a batch machine. Batch operation is assumed for max_ba > 1. If max_ba is set larger than 1000, then the maximum amount of **wafers** *nnn* in a batch is set by 1000 + *nnn*.

min_ba:
(1 <= min_ba <= max_ba): Minimum number of lots (requesting the same process step) which have to be in the queue of the tool set before process is allowed to start a batch (fixed threshold).
(min_ba < 0): Heuristic batching rule depending on the frequency of incoming lots and the process duration (variable threshold, not yet implemented).
(min_ba = 0, max_ba > 1): the program uses the following model to describe batch operation for this tool set:

******************************* simple batch model *********************************

Batch size > 1 is simulated by increasing the number of machines. At the same time the average process step duration will be increased because lots have to wait for each other to be "batched".

Batch size	multiplicator to machines	multiplicator to step duration
1	1	1
2	2	1.25
3	2	1
4	3	1.25
5	3	1
6	4	1.25
7	4	1
8	5	1.25

Step duration is given by the actual rec*nn*.
If this calculation results in more than 50 machines overall, the model does not work. A warning is issued.
****

If min_ba is chosen to be larger than 1000, than the minimum amount of wafers *nnn* needed to start a lot is set by 1000 + *nnn*. For example min_ba 1060 and max_ba 1150 states that in this toolset batch operation occurs with minimum of 60 wafers up to a maximum of 150 wafers. Only lots having the same process are assembled to a batch.

dispa

Each toolset has three buffers in front of the machines, one for Prio 1, one for Prio 2 and one for Prio 0 (standard) lots. Sorting of lots inside each buffer to control processing sequence is called dispatching. There are several dispatching rules available in FabSim. Dispa 0 - 5 rules evaluate lots in the buffer in front of the respective toolset. Dispa > 100 will allow customer defined rules taking the actual fab status into account (only FabSim Interactive). Setting column dispa to **0** yields the standard rule **FIFO** (first in, first out), sometimes called FCFS (first come, first serve). The lot which has arrived firstly in the buffer will be the first to be selected for processing. Dispa **1** selects the **CR** (critical ratio) rule. Generally lots with shorter remaining processing time are processed first. CR is calculated by (Due – Now)/(1 + TRPT). Due is the due date of the lot, Now the current time and TRPT is the total remaining processing time (according to Rose, WSC 2002). Dispa **2** selects the **SPTF** (shortest processing time first) rule. The lots in the buffer are sorted according to their process step times requested from the current toolset. The lot with the shortest process step time is selected for processing. If times are equal, the second sorting criterion is ‘ longest waiting time in the buffer’ first. There is a preset maximum waiting time (e.g. 720 min), then the waiting lot moves to the front of the buffer irrespectively of process step time. Dispa **3** selects the **ODD** (operation due date) rule. The ODD of operation i (process step i) is defined as lot start time plus ‘raw processing time RPT(i)’ times ‘flow factor’. RPT(i) denotes the RPT for a sequence of processing steps from step 1 to and including step i (see Rose, WSC 2003). Dispa **4** selects the **EDD** (earliest due date) rule. The lot with the due date nearest to the current time will be processed first. **WSPT** (weighted shortest processing time) is another dispatching rule available with Dispa **5**. It applies the lot attribute as the lot’s weight.

In FabSim Interactive the Dispa parameter will also select any customer defined dispatching which resides inside Cdispatch.dll. This is achieved by entering an integer value larger than 100 for Dispa, e.g. *mmmnn*. The last two digits *nn* select the internal dispatching as described above (currently 00 to 05). The upper digits *m*, *mm*, or *mmm* select the dispatch rule inside Cdispatch.dll. If for example you select 2203 for Dispa, the last two digits 03 will set ODD as the standard dispatching rule in the respective toolset. Every time a lot enters the buffer in front of this toolset, all lots inside the affected buffer queue are sorted according to the ODD rule. The other digits, here 22 will select a sorting rule which has to be defined in Cdispatch.dll. Sorting within Cdispatch.dll is started only just before a lot shall enter the machine. The sorting algorithm (sort for lists from STL) is stable, so the ODD sequence will be kept if the 22 sorting cannot discern between lots.

Currently there is a buffer sorting according to a priority given by the lot’s attribute (1 is the highest, a larger number a lower priority) if you select *n* as 1, another example dispatches lots according to the size of the following toolset buffers (*n* as 2). Thus Dispa **103** selects the new priority, if there are lots with same priority inside a buffer, they are kept sorted according to the internal rule Dispa3, ODD. Dispa **203** will select a sorting firstly according to ODD, then depending on the buffers of consecutive toolsets.

Each buffer (Prio 1, 2, 0) in front of the toolset is sorted individually according to the rule given. Generally lots in the Prio1 buffer are processed first, then lots in the Prio 2 buffer follow, finally standard lots (Prio 0) will be processed.

The command line parameters –all_cr, –all_sptf, -all_odd, -all_edd, -all_wspt or -all_dispa *mmnn* will set **all** toolsets to the respective dispatching rules, overwriting the selection in column dipsa.

load_d

time needed to load a lot into machine of respective toolset.

op_load

sets the operator group id and the operator time for loading the lot: If set to 0, no operator is needed. The integer value *[k]klmmm* is a combination of several data encoded into a single number. The first one or two digits *[k]k* denote the operator ID (value from 1 to 99). The center digit *l* , if set to 1, defines the following three digits *mmm* to set the absolute value of time needed by the operator (ranging from 001 to 999 minutes). If *l* is set to 2, the three digits *mmm* set the operator time requested to be *mmm*% of **load_d**. mmm may range from 001% to 999% (so may be set to a larger value than 100%). If mmm is set to 000, no operator is needed. Examples for **op_load** are: 21045 yields: ID is 2, 45 minutes are needed to load the lot, 232010 denotes: ID is 23, 10% of **load_d** is needed, 112120 denotes: ID is 11, 120% of **load_d** is needed. Independently of the operator time required, the lot suffers a load delay time of **load_d** minutes. Operator availability is always assumed. See chapter 3.4.10 (Operator data) and chapter 3.5.11 (Operators required) for more information on using operator data for factory simulation.

unl_d

time needed to unload a lot from machine of respective toolset.

op_unl

sets operator group id and operator time for unloading the lot. See **op_load** for further explanation.

transp

is a delay for each lot after leaving the specified machine. It simulates transport time. It will not add to the theoretical cycle time. An additional delay will result for batch machines because only one lot per minute will be returned for further processing. If ‘transp’ is set negative for any toolset, it’s value will be replaced by the corresponding value from the from-to matrix (see section 3.4.3).

op_tran

sets operator group id and operator time for transporting the lot. See **op_load** for further explanation.

mtbf, mttr:

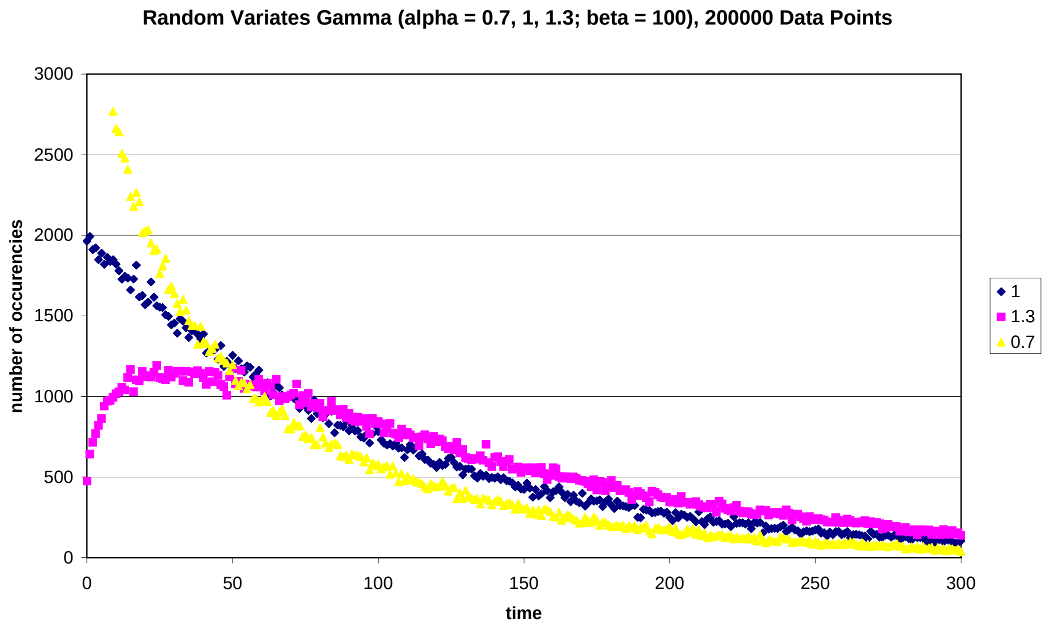
are used to describe machine uptime (mean time between failures and mean time to repair). All machines defined are o.k initially after start. The entered mtbf and mttr data are per default the mean values of two **exponential distributions**. Actual values of time to failure and repair time are determined by pseudo random number generators which generate random variates from the exponential distribution with given mean. If mttr = 0, the toolset with all it’s machines in the respective row of mach_dat.mset is assumed to be always defect free. If –d option is set on the command line, every downtime will be listed in fab_sim.log.

According to (Law, Kelton 2000) the exponential distribution is often not adequate to describe mtbf and mttr correctly, one better should choose the **gamma distribution** with two input parameters: the shape parameter *alpha* and the scale parameter *beta*. This is offered as an option in FabSim.

Selecting the command line parameters -ga_mtbf, -ga_mttr will preset all mtbf or mttr values to be drawn from the Gamma distribution. The shape factor alpha, then valid for all toolsets, may be entered on the command line as a real number following -ga_mtbf or -ga_mttr. mtbf and mttr from mach_dat.mset will deliver the scale factor beta.

mtbf and mttr can be changed locally for each toolset if entered as six or seven digit numbers in mach_dat.mset. If mtbf or mttr are entered equal or larger than 100000, FabSim will automatically assume the gamma distribution for the specific toolset. The entered number *aabbbbb* will be divided into two parts (*aa* and *bbbbb*). The last five digits (*bbbbb*) are reserved for the scale parameter *beta*, the first one or two digits (*aa*) for the shape parameter *alpha*. *alpha* will be set as *alpha* = a.a, *beta* will be set as *beta* = *bbbbb.* If mtbf is entered as 700400, then *alpha* = 0.7, *beta* = 400. If mttr is entered as 1303150, then *alpha* = 1.3, *beta* = 3150. If you enter 1000385, then *alpha* = 1.0, *beta* = 385. With *alpha* = 1.0 the gamma distribution simplifies to an exponential distribution with the mean 385. So entering the numbers 1000385 or 385 is equivalent, if Gamma is not selected by the command line parameter -ga_mttr. If for example you have set -ga_mttr 1.4 on the command line, you may overwrite this locally for any toolset to be exponential by setting mttr to e.g. 1000385, which yields *alpha* = 1.

The following plot compares the outputs of the gamma random variate generator with setting *alpha* = 1.3, 1.0, 0.7 and *beta* = 100. The "number of occurrences" on the y-axis denote how often an mtbf (or mttr) will occur with a duration set on the x-axis. Choosing *alpha* = 1.3 for mttr is reasonable, because now (according to the plot) a mttr of 0 occurs less often than say mttr = 40. A short mtbf occurs more often, when *alpha* = 0.7, compared to the exponential distribution.



In the absence of data, with the shape parameters given above, we need to determine the scale parameters for mtbf and mttr from machine engineering or vendor data. According to (Law, Kelton 2000) it is convenient and typically feasible to obtain an estimate of mean downtime $e = \mu_D$ and an estimate of machine efficiency *e*, as shown below.

$$e = \frac{\mu_B}{\mu_B + \mu_D}$$

μ_B is the mean amount of machine busy (processing) time before a failure. If the machine idle time (waiting for lots) is small, then μ_B equals the machine uptime. The scale parameters for mtbf and mttr then are given by

$$\beta_{mtbf} = \frac{e\mu_D}{0.7(1-e)}$$

and

$$\beta_{mttr} = \frac{\mu_D}{1.3}$$

op_mttr

sets operator group id and operator time for machine repair as absolute value or as percentage of **mttr**. See **op_load** for further explanation.

stbm1, sttm1, stbm2, sttm2:

describe a scheduled time between maintenance (stbm..) and duration of maintenance (sttm..) for two periods. You may chose fixed time based periods (e.g. weekly and monthly). Maximum time is 2000000 minutes. Repetition rate is stbm1 + sttm1 (stbm2 + sttm2). If there is more than one machine per tool set, the onset of maintenance is evenly distributed during the period stbm1 + sttm1. If scheduled and unscheduled downtimes overlap, they are simply added.
As an option you may chose a maintenance period determined by the number of processed batches or lots (e.g. maintenance of a LPCVD furnace after 52 lots). This option is activated by setting **stbm1** or **stbm2** to a value between 2000000 and 3000000, e.g. to 2000052. This will cause a maintenance activity of duration **sttm1** or **sttm2** after 52 lots have been processed since the last maintenance.
As another option you may chose a maintenance period determined by the number of processed wafers (e.g. maintenance of a metal etch tool every 1100 wafers). This option is activated by setting **stbm1** or **stbm2** to a value larger than 3000000, e.g. to 3001100. This will cause a maintenance activity of duration **sttm1** or **sttm2** every time 1100 wafers have been processed.
If **sttm1** is set to 0, no maintenance will be performed during the first period. If **sttm2** is set to 0, no maintenance will be performed during the second period.
If –d option is set on the command line, every maintenance action will be listed in fab_sim.log.

op_mt1, op_mt2:

sets operator group id and operator time for machine repair as absolute value or as percentage of **sttm1** or **sttm2**. See **op_load** for further explanation.

min_wt:

is used to model process step duration for lots with less than 25 wafers. If min_wt is an integer number from 1 to 24, the int_d and rec0 ... rec8 will be scaled by (number of wafers in lot)/25. The number of wafers is set in lot_sequence.strt for each lot. If the number of wafers started in a lot is less than min_wt, the process step time is multiplied by min_wt/25 and therefore does not scale further. If min_wt is less than 0 or larger than 25, int_d and rec*n* will not be scaled. Batch systems do not allow scaling of process duration.

setup:

Machines may need a setup before a lot can be processed. The duration of a setup procedure depends on the status of the machine. The status has typically been set by the lot which has been processed most recently. Only after the setup time has passed, processing of the next lot may start.

Type of setup, depending on the type of the machine:

0: no setup time for current toolset needed

1: always use setupt1 as setup delay time in current toolset

2: always use setupt2 as setup delay time in current toolset

11: setup for stepper
Setup time chosen is determined by comparing the previous and current product number and process step number. If both are the same, no setup time is required. If the product number is the same, but process step is different (corresponds to a reticle change within the same mask set), setupt1 will be chosen as setup delay time. If the product number changes (change of mask set), setupt2 will be chosen as setup delay time.

12: setup for implanter
Recipes are divided into three groups:
rec0-rec2 are recommended for B implant recipes
rec3-rec5 are recommended for P implant recipes
rec6-rec8 are recommended for As implant recipes
The recipe number used by the previous lot in each implanter machine of the toolset which has setup=12 will be stored. If the new lot's process step is going to implant the same ion type, that is its recipe number is from the same recipe group as the previous lot implanted (e.g. previous rec3, new rec4, both use P), then a short setup time is used (setupt1). If the new requested recipe is from a different group (i.e. the ion species has to be changed), then a long setup time (setup2) is used. Only after the setup time has passed, the lot will be started. Further setup types and models may be added here later.

21: setup for stepper according to setup avoidance rules
22: setup for implanter according to setup avoidance rules
Both setups follow the setup steps as already described above (**11**, **12**). However the lot selection rules try to avoid this setup if ever possible. If a single lot entering a toolset with empty buffer meets a single machine available, it will be started immediately, with setup, if necessary. If a single lot entering a toolset with empty buffer meets more than one machine available, it will check if one of the machines has already a suitable setup. If yes, it will choose this machine, if not, it will enter the first machine available. If the toolset buffer contains more than one lot (i.e. all machines have been occupied for a certain period) and a machine becomes available, the system will check through the buffer, starting with the front lot, if a lot waiting has a suitable setup requirement. If yes, this lot will be chosen for processing. If not, the first lot in the buffer will be chosen.

Priority 1 lots will always be handled first, then priority 2 lot are investigated and chosen.
The dispatching rules CR and SPTF may be used to determine the lot sequence in the buffer, before the setup avoidance procedure is started.
Setup avoidance is currently not available with batch systems.

op_set
sets operator group id and operator time for machine setup as absolute value or as percentage of **setupt1** or **setupt2**. See **op_load** for further explanation.

setupt1:
Setup time for short setup

setupt2:
Setup time for long setup

int_d:
The duration of recipes rec*XX* is determined by the throughput of the machine (given by the number of wafers per hour, e.g. 120 wafers/h in a litho cluster with coater, scanner and developer). This does not mean however that a specific lot will leave the machine rec n = 60min/120*25 = 12.5 minutes after it has entered. Wafers will see several sequential process steps (coating, exposure, developing) which add to the minimum time given above. So the machine will store the wafers for an additional time int_d, but allow other pending lots to enter. The total time of a lot in the machine is therefore rec*XX* + int_d. You may view it like this: a cassette of wafers is put onto the loading station of the cluster. One wafer after the other is loaded and runs through the internal process step sequence of the cluster. Meanwhile the next cassette with the next lot is started. The first cassette may be removed only after the last wafer of its lot is recollected from the cluster. Typically three or four lots are processed in parallel. int_d may

depend on the number of wafers in a specific lot. int_d adds to the theoretical process cycle time (calculated for a lot of 25 wafers). int_d is automatically set to 0 for batch systems. int_d may be scaled according to the number of wafers in a lot (see min_wt).

op_proc

sets operator group id and operator time for the process step performed as an absolute time value or as the percentage of the actual process step. See **op_load** for further explanation.

rec0 ... rec8:

set the duration (in minutes) of the respective recipe. It is the lot "takt" time, the inverse of the machine's maximum throughput. The lot process time is rec*XX* + int_d. The recipe duration is given for a lot of 25 wafers. It may be scaled according to the number of wafers in a lot (see min_wt). The command line parameter -rand_dur *%n* will randomly add (or subtract) a Gaussian variation to the duration, where *%n* is the standard deviation given in percent of the duration.

comment:

A character string not used by the program, may describe the process step and/or recipe name.

****

3.4.3 *** mach_dat.fto (from-to matrix) ***

According to the toolset data which are stored in mach_dat.mset you may define a file containing a matrix of transport delay times (default mach_dat.fto or the file name following the command line parameter –in_fto). This matrix contains transport time values for lots being transported between or within toolsets. If the parameter ‘transp’ in mach_dat.mset is set negative, FabSim will search for mach_dat.fto, look up the transport delay between toolset *mm* and toolset *nn* and replace with this value the negative value found in ‘mach_dat.mset’. If mach_dat.fto is not found, a default value of 5 will be used to replace the negative value.

```
from  to
   0  1     2     3     4     5     6     7     ...
0  5  11    11    11    11    11    11    11    ...
1  9  5     11    11    11    11    11    11    ...
2  9  9     5     11    11    11    11    11    ...
3  9  9     9     5     11    11    11    11    ...
4  9  9     9     9     5     11    11    11    ...
5  9  9     9     9     9     5     11    11    ...
6  9  9     9     9     9     9     5     11    ...
7  9  9     9     9     9     9     9     5     ...
...
...
```

The matrix is an ASCII file. Integer values are separated by white spaces. Following the title line, the second row display the toolset numbers. The first column also shows the toolset numbers. Therefore starting with column 2 and row 3 you may define a square matrix of size (no. of toolsets) x (no. of toolsets) with toolset 0 in the upper left corner. The diagonal line representing equal toolset number in row and column describes a transport delay within a toolset. The example sets a value of 5 min. for transport inside a toolset. If a lot moves from toolset 6 (look it up at the first column on the left) to toolset 2 (looked up on the row below the title), you will find a delay of 9 min.. If it moves from toolset 2 to toolset 5, transport will last for 11 min. Each time delay may be set individually, even asymmetrically (2 to 5 may differ from 5 to 2).

If an error occurs during reading mach_dat.fto, FabSim tries to replace missing data by a default value (preset to 5 min) and continues with a warning.

****

3.4.4 ****   mmm.proc (Run Sheets) ****

Run sheets are stored in ASCII files called mmm.proc. The run sheet name mmm has to be an integer number, like 1000 or 12. The files to store the run sheets are then called 1000.proc and 12.proc. The number of steps within each run sheet is limited to 700.

The example shows the header of a 1.2µm process flow chart:

```
12.proc
204 steps in run sheet
Number   Toolset  Recipe   Rework   Comment
0        13       0        0        Wafer mark
1         0       0        0        Clean
2         6       0        0        Pad oxide
3        15       0        0        Measure thickness
4        22       0        0        LPCVD nitride
5        29       0        0        HMDS
6        30       0        0        Coat resist
7        32       0        5        Stepper i-line
8        31       0        0        Develop
```

The run sheet file format is fixed as follows: The maximum number of characters in a row is 80 (including comments). Line 1 is a title and will be ignored. Line 2 begins with the number of steps followed by a blank or tab and a comment. Each of the following lines begins with a line number (will be ignored by the program), the machine number, the process step number, the rework probability (in percent) and a comment. All numbers and the comment are separated by a tab (maybe a blank). At least number, machine and recipe are required in each line. If you want to add a comment or step description, you will have to give a rework probability in the fourth column (may be 0) to fill the gap.

If the rework probability of a process step is set to *rw > 0*, the lot may suffer a rework with *rw %* probability. If a random number evenly distributed between 0 and 100 is less than rw, the lot flow control jumps into a rework flowchart (see section 3.4.8), runs through all rework process steps and then reenters the original flow chart in the step which immediately follows after the step which has generated the rework.

****

3.4.5 **** Processing time constraints ****

With some “magic numbers” in the toolset column of a run sheet (9999, 11111, 22222, 22223) you may influence the process step sequence of a lot. Processing time constraints are defined by the number couple 22222 (begin constraint), 22223 (end constraint). As shown in the example, the period of constrained time starts with row 96, by setting the toolset number to 22222. The recipe number given in this row (here set to 700) sets the maximum time allowed between start and end of the constrained period. The end of this period is defined by setting toolset no. to 22223 (example row 101). The other figures in this row are not relevant. To allow short processing in this section, the priority of the lot is raised by one level (and set back to the previous value after the section). A warning is issued if the requested time constraints cannot be fulfilled.

```
512.proc with constraint section
318 steps in run sheet
Number   Toolset Recipe   Rework   Comment
...
95       0       0        0        Preclean
96       22222   700      0        Begin time constraint 700 minutes (RPT 410)
97       1       0        0        Clean HF
98       8       0        0        Gate oxide
99       15      0        0        Measure thickness
100      23      1        0        Poly deposition
101      22223   0        0        End time constraint
102      15      0        0        Measure thickness
```

```
...
```

****

3.4.6 **** Alternative toolset selection ****

Two or more alternative toolsets may be offered following after the row with the toolset no. 11111 (row 8 in the following example). The “recipe”-value given in this row sets the number of alternative toolsets. The value of 2 given in the example below sets the following 2 rows to describing alternatively selectable toolsets. FabSim then selects the toolset out of the two with the least amount of lots pending in its toolset buffer.

```
512.proc with alternative toolsets
318 steps in run sheet
Number  Toolset Recipe  Rework  Comment
...
7       29      0       0       HMDS
8       11111   2       0       2 alternative coaters
9       30      0       0       Coater 1
10      31      0       0       Coater 2
11      32      0       0       Stepper i-line
12      31      0       0       Develop
13      19      0       5       Visual inspection
...
```

****

3.4.7 **** mmm.proc (Run Sheets for test lots) ****

You may define test lots with their own run sheets according to the rules for normal lots defined above. The difference is that the final step to be processed from this run sheet has to be followed by a step containing 9999 as the toolset number. Within the file mmm.proc the machine number 9999 may appear as the last step or somewhere in between. These test lots will be processed according to the run sheet in normal fashion until 9999 is reached. At this point the lot is “scrapped”. The resulting lot data will not be stored in the output files fab_sim.out or fab_sim.exo, but will be collected as a scrap lot in fab_sim.scp. During processing however these lots will use machine resources and add to the wip count.

Example (Header of a 1.2µm process, lot crashes in step 6):

```
12.proc  crash in step 6
204 steps in run sheet
Number  Toolset  Recipe  Rework  Comment
0       13       0       0
1        0       0       0
2        6       0       0
3       15       0       0
4       22       0       0
5       29       0       0
6     9999       0       0       lot crashes
7       32       0       5       Stepper i-line
8       31       0       0
```

****

3.4.8 **** mmm.proc (Run Sheets for rework) ****

In the run sheet 12.proc shown above, step 7 may suffer a rework with 5% probability. If this rework has to be done (here: process 12.proc, tool set 32) the program will ask for a rework run sheet 12032.proc (which is shown below). The normal process flow is interrupted, the rework run sheet will be processed, the normal process flow resumes. You will have to provide this run sheet to allow rework for each tool set where the process run sheet's parameter "rework" is larger than 0. It's name is constructed by 1000*orig_proc_name + tool_set_number, here: 1000*12+32, where orig_proc_name is the name of the original run sheet used by the lot which suffers rework. tool_set_number is the tool set where rework occurs (better: where it is detected, e.g. litho rework necessity is detected during visual inspection). Inside a rework run sheet there is further rework allowed. To simplify the model, i.e. not to increase the storage depth of all lot parameters, a rework inside a rework process will simply restart the rework from the beginning.

In the example rework flow chart below litho errors may be detected with a rework probability of 8% during the inspection step (toolset 19). Thus in case of rework required, the rework process flow is repeated from step 0.

```
12.proc  Litho rework toolset 19
8 steps in run sheet
Number   Toolset Recipe  Rework  Comment
0        26      0       0       Resist strip
1        3       0       0       Posistrip clean
2        29      0       0       HMDS
3        30      0       0       Coat
4        32      0       0       Stepper i-line
5        31      0       0       Develop
6        19      0       8       Visual inspection
7        18      0       0       CD
```

****

3.4.9 **** mmm.proc (Run Sheets for Send Ahead Wafers) ****

The rework capability may be used to emulate send ahead wafers. If the rework percentage is set to equal or larger than 100, the rework sequence will always start. Typically the sequence, however, commences after the process step has been performed. So a send ahead wafer has to be defined in the line before the process step which has to be tested by a send ahead wafer. In the example a send ahead for the stepper is defined in line 6, a rework (with 5% probability) in line 7. The Rework column value 101 will send 1 wafer into the send ahead loop (105 would send 5 wafers). A “send ahead”-flow chart 12030.proc has to be provided (as well as a rework flow chart 12032.proc).

```
12.proc  send ahead in line 6, rework in line 7
204 steps in run sheet
Number   Toolset  Recipe  Rework  Comment
0        13       0       0
1         0       0       0
2         6       0       0
3        15       0       0
4        22       0       0
5        29       0       0
6        30       0       101     one send ahead wafer
7        32       0       5       Stepper i-line
8        31       0       0
```

****

3.4.10 **** operators.in (Operator data) ****

To calculate the needed amount of operators, some basic input data are required. These are read from *operators.in* (or the file with filename entered on the command line after -in_op (see chapter 3.3).

```
Operator data 05/02/04
Working hours per week (or per seven days)
40
Holidays (weeks per year)
5.8
Percent ill or absent
4.5
Time offset
2
************************
```

‘Working hours per week’ (real number) denote the average working hours of an operator per week (with 52.15 weeks per year). ‘Holidays’ (real number) are the average weeks off per year during holidays. ‘Percent ill or absent’ (real number) gives an estimate of ‘other causes’ of non-work time. ‘ Time offset’ (integer number) adds a fixed offset to all operator times (e.g. for operator movement from one station to another).

3.5 Output Files

The output file names may be set by the -out xxx command line parameter to replace the predefined file names. Already existing output files with same name will be overridden without warning. Information on where to locate input and output files is given in section 3.4.0.

3.5.1 ***** fab_sim.use (or xxx.use) *****

Buffer and machine usage   Wed Jan 28 00:45:43 2004

```
toolset  mean    mean     average   per machine
number   wait  occupancy  usage[%]  usage[%]
  0       283     3.6       50        50
  1        13     0.0        6         6
  2       100     1.3       47        47
  3       172     0.7       25        25
  4       155     0.1        5         5
  5       487     1.6       59        59
  6       243     0.6       44        44
 32       133     1.3       42        47       38
```

The usage of each toolset is calculated between use time2 (which has been set by [-use time2] on the command line) and the end of the simulation. “mean wait” gives the average waiting time of a lot in the toolset buffer. "mean occupancy" calculates the time average occupancy of the toolset buffer (in number of lots stored). "average usage" displays the usage (lot(s) in process) in percent of the simulation time averaged over all machines within the toolset. "per machine usage" prints the usage (in %) of each machine.

As an example toolsets 1 – 6 and 32 are shown. Toolset 32 has two (equal) machines, an average of 1.3 lots are stored its buffer, average usage is 42%, individual machine usages are 47% and 38%.

You may evaluate this file under MS EXCEL with the spreadsheet (containing proper macros and filters) EVAL_use.xls.

3.5.2 ***** fab_sim.out (xxx.out) *****

This file is the "output" of the fab. Each lot which has been processed completely is listed here in the sequence of its arrival at fab-out.

```
lot:  proc: prod.:  wafers: prio: start   stop:    dur:
2      12     1       25     0    2400    24718   22318
...
216   5121    1       25     0    259200  356078  96878
...
```

A summary of simulation data is listed at the bottom of the file:

```
Process 512:  theoretical cycle time 28145 minutes
Process 12:   theoretical cycle time 12410 minutes
Process 5121: theoretical cycle time 32675 minutes

Lots processed:  65

Start:    Sun Sep 30 12:13:40 2001
End:      Sun Sep 30 12:14:13 2001
Simulation time: 33 sec
```

The command line parameter –no_out will suppress writing to this file. After the simulation is finished however, all output data are written to fab_sim.exo. You may evaluate these data under MS Windows with the EXCEL spreadsheet (containing proper VBA macros) EVAL_ct.xls.

3.5.3 ***** fab_sim.exo (xxx.exo) *****

A modified output file is created here. It is formatted to give optimum input to MS EXCEL. Data are sorted according to the processes used. You may evaluate this file under MS Windows with the spreadsheets (containing proper macros) EVAL_ct.xls (cycle time vs. time), EVAL_proc_t.xls (lots out per selected period, sorted by processes).

3.5.4 ***** fab_sim.buf (xxx.buf) *****

This file shows the amount of lots waiting in the buffer in front of each tool set versus simulation time. It may be displayed and analyzed with the EXCEL spreadsheet EVAL_buf.xls. FabSim generates this file with the '-trace' command line option. Trace files may become very large. Therefore '-trace *nn* ' reduces trace file size by limiting tracing to toolset no. *nn*.

3.5.5 ***** fab_sim.wip (xxx.wip) *****

This file displays (versus simulation time) the number of lots active in the factory, also called "work in progress" or "wip". When a lot enters the fab, wip count is increased by 1, when it leaves (ready or scrapped) wip is decreased again. You may display and analyze the file with EVAL_wip.xls. FabSim generates this file with the '-trace' command line option.

3.5.6 ***** fab_sim.log (xxx.log) *****

This (usually large) file is created with the '-d' command line option. It lists each process step immediately after its start. In addition defect machines are posted, as well as Prio 1 lots having retarded other lots. You may use this file for debugging purposes.

Parameters listed are: actual time, machine, system within machine, step duration, lot number, start time of lot, process selected, step number in flow chart.
time: 2014, mach: 15, subs: 0, dura: 22, lot: 1, start: 1200, proc: 512, step: 5
There will also be information listed on failed machines and Prio 1 lots.

You may evaluate this file under MS EXCEL with the spreadsheet (containing proper macros and filters) EVAL_fablog.xls.

3.5.7 ***** fab_sim.scp (xxx.scp) *****

Test lots whose run sheets contain 9999 as a machine number will appear here after being processed when they have reached the 9999 breakpoint.

3.5.8 ***** fab_sim.tra (xxx.tra) *****

This file shows the amount of machines available (i.e. not defect) in each tool set versus simulation time. It may be displayed and analyzed with the EXCEL spreadsheet EVAL_tra.xls. FabSim generates this file with the '-trace' command line option. Trace files may become very large. Therefore '-trace *nn*' reduces the trace file size by limiting tracing to toolset no. *nn*.

3.5.9 ***** fab_sim.occ (xxx.occ) *****

This file shows the amount of machines in each tool set currently processing wafers versus simulation time. It may be displayed and analyzed with the EXCEL spreadsheet EVAL_occ.xls. FabSim generates this file with the '-trace' command line option. Trace files may become very large. Therefore 'trace *nn*' reduces the trace file size by limiting tracing to toolset no. *nn*.

3.5.10 ***** fab_sim.prod (xxx.prod) *****

A modified output file is created here. It is formatted to give optimum input to MS EXCEL. Data are sorted according to the products fabricated. You may evaluate this file under MS Windows with the spreadsheet (containing proper macros) EVAL_prod_t.xls (lots out per selected period, sorted by products).

3.5.11 ***** fab_sim.opo (xxx.opo) *****

Operators required for the specific simulation are listed here. The input file mach_dat.mset enters operator times for any activity in the fab. During simulation these values are accumulated according to the process steps performed. During final output the data are weighted by input data from operators.in and stored in fab_sim.opo (*.opo). The following simple model is used to calculate the number of operators needed for each operator id:

$$op.time.py = (weeks.py - holidays.py) * (1 - 0.01 * percent.ill) * working.hours.pw * 60$$

$$operator.available.time = sim.period \div time.py * op.time.py$$

$$number.of.operators = \frac{operators.needed.time}{operator.available.time}$$

3.5.12 ***** fab_sim_*nn*.fstat (xxx_*nn*.fstat) *****

Saves the current factory status into a text file. *nn* is the simulation time at the time of saving the factory status data. So during a single simulation run a dedicated file may be generated at each simulation stop.

For each toolset the total number of tools is listed, followed by the tools currently in use, the tools down for scheduled or unscheduled maintenance, the number of lots waiting in the buffer and the number of tool setups since simulation start.

```
Factory status after 200000 minutes:

WIP is 920 lots.

toolset tools   in use  down    buffer  setups
0       11      3       0       0       0
1       3       1       0       0       0
2       9       6       1       0       0
3       6       2       1       0       0
4       3       2       1       0       0
5       13      9       2       0       0
6       5       3       0      10       135
...
...
```

3.5.13 ***** fab_sim_*nn*.lstat (xxx_*nn*.lstat) *****

Saves the actual lot status into a text file. *nn* is the simulation time at the time of saving the lot status data. Lots may stay inside a toolset or are ready or scrapped

`Step` lists the current process step (read from the flow chart) of each lot. `Toolset` gives the toolset where the lot is to be found at data saving time or if it is ready or scrapped. A rework may currently occur, then the rework flow chart is given in `Rework`.

This file may be read into FabSim (see LoadData) to start simulation with a given lot status.

```
Lot status after simulating for
200000
minutes:

Lot     Start   Prio    Process Step    Toolset Flow F. Rework
1       70      0       8       314     ready   3       0
2       140     0       8       314     ready   3       0
3       210     0       8       314     ready   3       0
4       280     0       8       314     ready   3       0
5       350     0       8       314     ready   3       0
6       420     0       8       314     ready   3       0
7       490     0       8       314     ready   3       0
8       560     0       8       314     ready   3       0
9       630     0       8       314     ready   3       0
...
...
1882    186700  0       8       116     19      3       0
1883    186800  0       8       2       29      3       8032
1884    186900  0       8       113     30      3       0
1885    187000  0       8       116     19      3       0
1886    187100  0       8       116     19      3       0
1887    187200  0       8       113     30      3       0
1888    187300  0       8       113     30      3       0
```

```
1889    187400  0       8       107     8       3       0
1890    187500  0       8       107     8       3       0
1891    187600  0       8       105     0       3       0
1892    187700  0       8       101     2       3       0
1893    187800  0       8       100     19      3       0
1894    187900  0       8       100     19      3       0
...
...
```

3.5.14 ***** fab_sim_*nn*.pstat (xxx_*nn*.pstat) *****

Saves the current lot status for a given product into a text file. *nn* is the simulation time at the time of saving the product status data. Lots may stay inside a toolset or are ready or scrapped. A rework may currently occur, then the rework flow chart is given.

```
Lot status with product no. 1 after simulating for 300000 minutes:

Lot     Start   Prio    Process Step    Toolset Flow F. Rework
1       1200    0       512     314     ready   6       0
4       4800    0       512     314     ready   6       0
7       8400    0       512     314     ready   6       0
...
...
97      123600  0       512     314     ready   6       0
100     127200  0       512     314     ready   6       0
103     132000  1       512     314     ready   2.5     0
106     135600  0       512     312     36      6       0
109     139200  0       512     312     36      6       0
112     142800  0       512     311     36      6       0
115     146400  0       512     300     30      6       0
118     151200  1       512     314     ready   2.5     0
121     154800  0       512     285     30      6       0
```

3.6 Data Evaluation

Several data evaluation tools are distributed with FabSim. They are based on MS EXCEL. Therefore currently they are useful only in the MS Windows operating system. The *.xls files in the distribution contain proper macros to read FabSim output files and display the data.

3.6.1 EVAL_use.xls

Displays the *.use file graphically in a bar chart. Average usage of each tool set is plotted versus tool set number. Stores table and graph in a file fab_use.xls for later retrieval.

To start plot, double click on EVAL_use.xls in Windows Explorer. Select file from file list box.

3.6.2 EVAL_ct.xls

EVAL_ct.xls plots lot cycle time versus lot start time (or lot out time as option). It uses the file *.exo which contains lot output data similar to *.out, but in an EXCEL readable format.

Start with double clicking EVAL_ct.xls in Windows Explorer. Select file from file list box. Data will be plotted on worksheet “Plot”. Mean values and standard deviations of cycle time and tardiness for each process are tabulated in worksheet “Eval. Data”. These tables collect lots and data only after the usage time has passed. Graph and data may be stored by selecting “file, save as” from EXCEL menu, and then choose an appropriate file name. Tardiness is defined as *max*(*start* + *duration* - *due date*, 0). *Due date* is calculated by the product of flow factor and theoretical cycle time.

To display another file or change the option lot start time or lot out time, return to start window by selecting “Window, EVAL_ct.xls” from the EXCEL menu.

3.6.3 EVAL_proc_t.xls

EVAL_proc_t.xls displays a bar graph plot of wafers which left the factory, grouped into time periods (day, week, month, year, user selectable) and processes. It uses the file *.exo which contains lot output data similar to *.out, but in an EXCEL readable format.

Start with double clicking EVAL_proc_t.xls in Windows Explorer. Select file from file list box. Data will be plotted in columns (weekly output) on the worksheet “Plot”. Mean values and standard deviations for each process are tabulated in worksheet “Eval. Data”. Graph and data may be stored by selecting “file, save as” from EXCEL menu, and then choose an appropriate file name.

To display another file or change the time period, return to start window by selecting “Window, EVAL_proc_t.xls” from the EXCEL menu. You may select a period from the combo box or type in any integer value (period in minutes).

3.6.3 EVAL_prod_t.xls

EVAL_prod_t.xls displays a bar graph plot of wafers which left the factory, grouped into time periods (day, week, month, year, user selectable) and products. It uses the file *.exo which contains lot output data similar to *.out, but in an EXCEL readable format.

Start with double clicking EVAL_prod_t.xls in Windows Explorer. Select file from file list box. Data will be plotted in columns (weekly output) on the worksheet “Plot”. Mean values and standard deviations for each process are tabulated in worksheet “Eval. Data”. Graph and data may be stored by selecting “file, save as” from EXCEL menu, and then choose an appropriate file name.

To display another file or change the time period, return to start window by selecting “Window, EVAL_prod_t.xls” from the EXCEL menu. You may select a period from the pull down menu of the combo box or type in any integer value (period in minutes).

3.6.4 EVAL_fablog.xls

Lists *.log file (created with –d [number] option) in an EXCEL table and activates the automatic filters. This table may be used to analyse the detailed log data. You may choose entries according to tool set, lot, process or step number. Stores data in fab_log.xls.

To start table, double click on EVAL_log.xls in Windows Explorer. Select file from file list box.

3.6.5 EVAL_lot.xls

Displays process step number versus simulation time for a selected lot. Data are read from the *.log file. Due to limitations of EXCEL (no more than 65635 lines per worksheet, exceeded when *.log is larger than about 7 MB) a temporary file lot.tmp will be created containing data only from the selected lot.

To start plot, double click on EVAL_lot.xls in Windows Explorer. Select *.log file from file list box, then enter lot number *nn*. *.log should be created by the plain –d command, that is without optional toolset number.

Process steps occurring in batch machines will not be plotted due to different numbering schemes in *.log.

3.6.6 EVAL_wip.xls

Displays *.wip file graphically as a scatter point diagram, showing work in progress (wip) versus simulation time.

To start plot, double click on EVAL_wip.xls in Windows Explorer. Select file from file list box.

3.6.7 EVAL_buf.xls

Displays *.buf file graphically as a scatter line diagram, showing lots in buffer of toolset *nn* versus simulation time.

To start plot, double click on EVAL_buf.xls in Windows Explorer. Select file from file list box. Select toolset (enter number and hit Return). Select modulo *xx* if amount of data is to large (only every *xx*th data point will be plotted.). Begin and End entries mark the time axis limits. Hit Return after each value entry. Then click Start for data display.

3.7 Tips, Comments

3.7.1 Running a single lot (FabSim, FabSim Interactive)

If you run a single lot through the fab (good for testing the machine data) keep in mind to give it priority 1 and -p1 option or set all minimum batch sizes (min_ba) to 1. Alternatively set –batch_wait to a reasonable time (e.g. 1000). Otherwise the lot gets stuck in the batch queue waiting for other lots to fulfill the min_ba requirement.

3.7.2 Check for maximum capacity (FabSim, FabSim Interactive)

The following input file may be used to check for the maximum capacity of a fab in a single simulation run:

```
title: 0.25um, fab throughput test
relative
number: proc: prod: wafers: start: prio:
start_loop
1       025   1     25      90     1
repeat 2222
start_loop
1       025   1     25      80     1
repeat 2500
start_loop
1       025   1     25      70     1
repeat 2857
start_loop
1       025   1     25      60     1
repeat
```

The lot start period will be shortened in each successive loop, the maximum capacity is reached when the cycle time will not stabilize for a given start rate, but will increase steadily.

3.7.3 Simulate influence of increase or decrease in machine count (FabSim Interactive)

To simulate increase in machine count per toolset in interactive mode, start FabSim und run for a few minutes, then set down the machines to be added later (by using *Machine Downtime*) and continue FabSim. Downtime should be larger than the stabilization time of the simulation. When the downtime is over, the machines will be added automatically to their respective toolsets. This emulates the addition of machines to the toolset. Decrease in machine count may be simply emulated by setting down the machines at a selected stop for a very long downtime.

3.7.4 FabSim Interactive dies without warning (FabSim Interactive)

Check file console.log for possible causes (missing files, demo version vs. full version ...).

3.7.5 Start lots in groups (FabSim, FabSim Interactive)

The following input file may be used to start lots in groups (here of three lots):

```
title: 0.25um, group lot start
absolute
number: proc:   prod:   wafers: start:  prio:   flowf:
```

```
1       512     1       25      5       0       0
start_loop
1       512     1       25      5       0       0
2       512     1       25      10      0       0
3       512     1       25      360     0       0
repeat
```

Lot starts occur at 5, 10, 15, 365, 370, 375, 725, 730, 735, 1085, 1090, 1095... Three lots are started every 360 minutes, giving 300 wafer starts per day.

3.7.6 Tool dedication

In a larger factory tool dedication may be an adequate means to improve throughput by avoiding setups entirely. A tool may be dedicated to run only one specific process step, two or more specific process steps out of a larger group of steps, be dedicated to run implants only for one atom species (e.g. only boron), or be dedicated to a specific process or product flow only (e.g. no change in shot map setup of a stepper). Whereas it might have been possible to integrate a specific program flow into FabSim for tool dedication, all the dedication examples are better handled by choosing adequate toolsets and recipes.

For example you may add three medium current implanter toolsets, one for B, one for As, one for P, each having one or more tools. If the setting within the flow charts is then done correctly, only the atom specific toolset is selected.

4 Literature and Links

4.1 Literature

H. Vogt: "Discrete-Event Simulation using SystemC: Interactive Semiconductor Factory Modeling with FabSim". In: Proceedings of the 2003 Winter Simulation Conference, ed. S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, 1383-1387. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers. Available online via http://www.informs-cs.org/wsc03papers/174.pdf

H. Vogt: “A New Method to Determine the Tool Count of a Semiconductor Factory Using FabSim“, In: Proceedings of the 2004 Winter Simulation Conference, ed. R .G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, 1925 - 1929. Available online via http://www.informs-sim.org/wsc04papers/256.pdf

W. J. Hopp, M. L. Spearman: "Factory Physics", second ed., McGraw Hill, Boston, 2000

G. S. Fishman: "Discrete-event simulation", Springer, New York, 2001

A. M. Law and W. D. Kelton: "Simulation Modeling and Analysis", McGraw-Hill, Boston, 2000

J. Banks, editor: "Handbook of simulation", Wiley, New York, 1998

4.2 Links

ASMC (Advanced semiconductor manufacturing conference) http://wps2a.semi.org

WSC (Winter simulation conference) http://www.wintersim.org/

ISSM (International Symposium on Semiconductor Manufacturing) http://www.issm.com/

Bibliography:
http://www.jkrconsult.com/qbib.htm and http://www.jkrconsult.com/capbib.htm